51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年12月
总第021期

12月编程语言排行榜：Objective-C风暴来袭

旅游会是开发者的另一片需求蓝海？

淘宝王琤：Taobao JVM的性能优势与价值体现

目录 2012.12 Contents

51CTO.com 技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡　林师授　张伟　封面设计：冯海舰

联系方法：linss@51cto.com　010-68476606（分机 8123）

出版日期：2012 年 12 月 21 日

欢迎来稿，请发送邮件至 linss@51cto.com

编者按

2012 年 12 月 TIOBE 编程语言排行榜又出炉了，这次的排行和上个月没有太大的变化。也许在你的意料之中，Objective-C 不断的继续上升。

2012年12月编程语言排行榜：Objective-C风暴来袭

【51CTO 独家特稿】2012 年 12 月 TIOBE 编程语言排行榜又出炉了，这次的排行和上个月没有太大的变化。也许在你的意料之中，Objective-C 不断的继续上升。而其他主流的移动应用编程语言，如 C，C++ 和 Java 都在上涨，但速度缓慢，很明显不能与 Objective-C 竞争。

从 Web 编程语言来看，Python 与 Ruby 也是一直属于增长的状态，不过提升相对来说比较慢。但值得关注的是以这样的趋势，不久会追赶上 PHP 的王者地位，甚至超越。

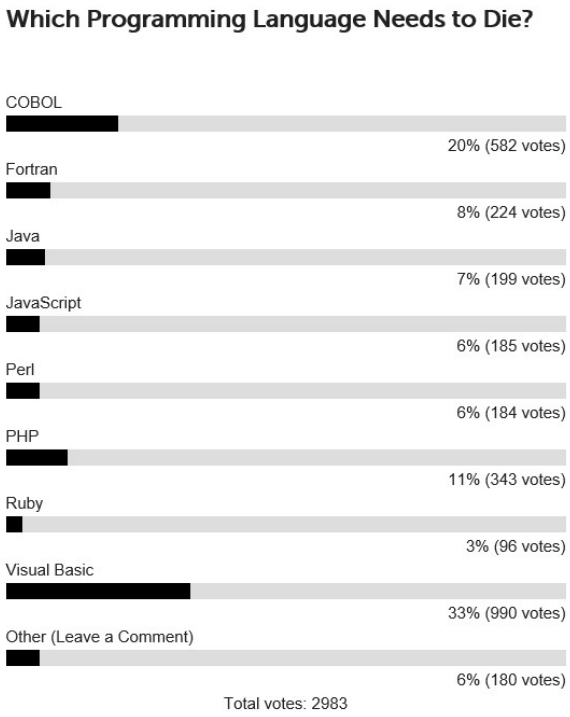
2012 年 12 月编程语言排行榜榜单

Position Dec 2012	Position Dec 2011	Delta in Position	Programming Language	Ratings Dec 2012	Delta Dec 2011	Status
1	2	↑	C	18.696%	+1.64%	A
2	1	↓	Java	17.567%	+0.01%	A
3	5	↑↑	Objective-C	11.116%	+4.31%	A
4	3	↓	C++	9.203%	+0.95%	A
5	4	↓	C#	5.547%	-2.66%	A
6	6	=	PHP	5.541%	-0.46%	A
7	7	=	(Visual) Basic	5.174%	+0.42%	A
8	8	=	Python	3.848%	+0.36%	A
9	9	=	Perl	2.174%	-0.30%	A
10	11	↑	Ruby	1.728%	+0.23%	A
11	10	↓	JavaScript	1.321%	-0.88%	A
12	12	=	Delphi/Object Pascal	0.977%	-0.27%	A
13	13	=	Lisp	0.949%	-0.23%	A
14	16	↑↑	Pascal	0.894%	+0.16%	A
15	35	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.889%	+0.53%	A
16	17	↑	Ada	0.648%	+0.02%	B
17	22	↑↑↑↑↑	MATLAB	0.608%	+0.07%	B
18	21	↑↑↑	Lua	0.601%	+0.05%	A--
19	19	=	Assembly	0.580%	+0.02%	B
20	14	↓↓↓↓↓↓	PL/SQL	0.574%	-0.23%	B

Ruby 前景分析

随着计算机的快速发展，编程语言也越来越多，在 10 年因为开发速度的问题，Java 成了编程语言的老大，随着 Java 不断的臃肿与复杂，开发者也不断的去寻找，尝试各种新的语言，也因近年动态语言有后来居上的趋势，Ruby 作为具有动态类型的解释型面向对象语言因此而被开发者热捧。革命的年代已经结束。Ruby 从 1995 年至今已经成熟成为了编程界的主流语言，至少主流在向 Ruby 前进。

当然，在这个充满偏见的语言战争年代，或许是一场无意义的比拼，这是哪个语言都无法回避的。从复杂性问题本身来考虑，为什么说它受到开发者的热捧？首先我们来张国外某博客发起的一次编程语言的投票。



■本文未完，详细部分请参考原文：

http://developer.51cto.com/art/201212/369470.htm

人类的想像力一向是无穷的。

随着 12 月 21 号越来越近，人们的末日恐惧心理也越来越严重。在玛雅预言中，12 月 21 日是世界末日，那天地球陷入一片黑暗。人们为了求生只好提前造好诺亚方舟，逃离即将毁灭的地球。而且网络上有越来越多的谣言，如 12 月 21 号将全世界黑暗三天，更是让很多人恐惧惊慌，导致出现了很多让人忍俊不禁的笑话来。

如有一网友将自己百万的家产全部捐出，以希望自己的善心能帮助自己渡过世界末日，现又有一女白领辞掉优越的工作，去山野学习原始生存方法，在深山里还住上洞穴，捉起鸟兽，过着野人般的生活。这都在网络上引起了轰动，有关心的，有鼓励的，有想跟随的，也有劝慰的。

2012 年 12 月 21 日已经来临，世界并未发生任何末日征兆，人类生活一切正常进行。危机就此解除，但同时给我们带来很多值得反思的问题，我们应该相信科学，还有就是多去关爱身边的人。

51CTO 开发频道寄语

编者按

Java 没死，事实上它拥有足够的能量让你的应用跑起来。那些对 Java 吹毛求疵人频繁地聚焦在一些小众问题上，总是和其他技术或者语言做些不公平的对比，这些语言并没有像 Java 一样得到广泛应用……

Java已死？九百万程序员说不！

Java 没死，事实上它拥有足够的能量让你的应用跑起来。那些对 Java 吹毛求疵人频繁地聚焦在一些小众问题上，总是和其他技术或者语言做些不公平的对比，这些语言并没有像 Java 一样得到广泛应用及长远的历史。

现在的小孩都能学 Java，它在 Web 和企业开发中广泛被采用，特别是近些年更是有了些让人惊讶的改善，一些新特性正在筹备中。即使抛开这些最新的特性，Java 仍然很酷，应用的广泛性、JVM 平台背后卓越的设计、清晰的语法、拥有丰富的工具和库文件构成的生态系统。Oracle 说有超过 9,000,000 Java 开发者（亿万的应用和设备用户），所以为什么我会听到诸如“Java 正在被淘汰中”，2007 年开始，Java 就已经沦为“21 世纪的 Cobol 语言”的言论？

Java 平台是工程师的梦想

首先就是 Java 平台的存在，HotSpot JVM 是一项非凡的设计，CLR（公共语言运行库）有了大量的优化，Java 应用在性能上甚至可以与 C 相媲美，当然还有其他可选的虚拟机供可用（如：JRockit,Zing），应你的环境有特殊的需求而定。

再者，多种基于 JVM 的语言使得这个平台更奇妙，如 :Groovy，Jython，JavaFX，Scala，当然还不局限于以上流行的语言。 Java 现在包含有字节码指令 :invokedynamic 和 java.lang.invoke 包，使得 JVM 构建动态语言更简单，现在已经有超过 50 中基于 JVM 的语言。其中最有意思的一个是 php.reboot，它的目标是保持 php 的哲学理念，但是移除了其中一些缺陷，而且它同样能在 Android 中运行。

Java 是一门成熟的语言，并不适合“老人”

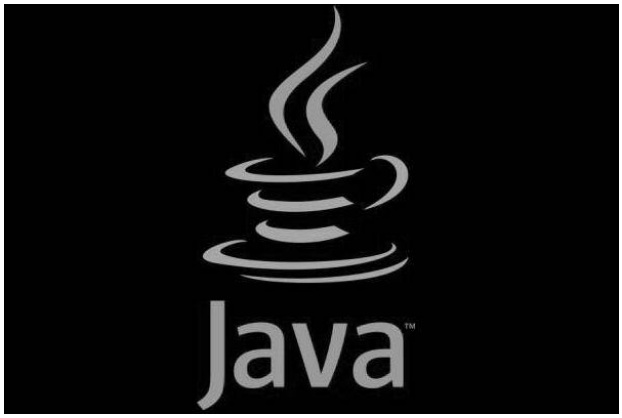
Java 成为被批评，抱怨，诅咒的靶子，我说这门语言还没死，恰恰相反当有人抱怨 Java 的时候正是推广 Java 时机。人们总会做些奇怪的对比，好像仍然以为 Java 还是停留在 1.4 的版本，用记事本编写、需要 EJB2 的支持、只能用来写写简单的留言板，然后把这些用来与一个高端框架甚至是 CMS 来对比。

作为 Java 程序员，这种对比对我来说没有任何意义。更为明智的方法对比 Java 就得理智地选择竞争对手，看看 Java vs PHP、Python 或者 Ruby，或者用框架 Play 对比 Ruby on Rails，SpringMVC vs Zend Framework，有鉴于此，我觉得 Java 似乎根本就不适合老人。

Java 冗长吗？当然

人们常说 Java 太罗嗦了，减缓了运行速度。批评者矛头通常指向 Java 的强类型静态语言特性，缺乏前沿技术在语言中，然而，我认为他们是经过深思熟虑的，而且这正是 Java 良好的特性。动态语言在启动一个小项目时开始会觉得很受欢迎，但是可以考虑一下，在时髦框架及合适的工具（如 :考虑用 IDE 代替记事本）下创建一个“Hello 留言板”类型的应用，Java 很简单，只是 10 分钟的事情，如果你想做个试验的话，使用 Spring Roo 再准备一个秒表，如果有需要的话。现在就可以开始繁琐的 CRUD 了。

想象一下，你正为移动运营商搭建一个系统，运行客户登陆网站，你不得不在后台调用各种子系统收集大量的数据，Cool 框架通常分解你的程序模块而不需要匹配任何用户模型。要更多的了解建议你看看 Joel Spolsky 发表的一篇文章。



Java 是一门强类型静态语言

强类型静态语言有很多好处，我喜欢它简单的视觉外观，我可以粗略看一段代码就知道是做什么的，它就像是可视化的用英语反馈，语言非常易读，字母混合成单词仍然可读。其它的一些好处是有强大的 IDE 支持，动态语言在这点一直是弊端，大项目中有强大的 IDE 和工具支持是无价的。

批判者的观点在 Java 在读文件、转换 xml 或迭代集合的时候缺乏表达力，但是你可以总是创建一个方法去处理这些常用的事例，或者是用 FileUtils.readLines()，java 语言上表达力的缺点有很多库对其支持。在 Java7 中能看到一些优雅的加强型功能，如自动关闭资源、siwtch 语句支持 String、数字类型支持下划线（强烈建议读 Coin 项目）。Java8 承诺了更多的东西（最有意思的应该是闭包）。Java 是不是在任何方面都是那么的完美无缺呢？当然不是，这就是 Java8、Java9 在准备阶段的原因。我个人也不喜欢那些不太优雅的核心 API. 是否的确将更多的事情留给平台而不是语言本身 ?java 核心包括 API 进阶设计跨度长达 20 年，API 的更新会破坏向后兼容性，有些设计太抽象，但有些还不够抽象，有些太零碎，有些根本不可思议。看看其竞争对手 .NET，核心 API 做的很好，比如统一的通信 API。Java8 在 Jigsaw 项目的帮助下，会有所改变。

所以你值得拥有 Java，正确地使用它，它是一门很棒的语言。堪比 Klingon 语言，它将继续改善并不会很快消失。不应把努力放在取代 Java 上，而是和其他 JVM 语言结合使用，这是最明智的，但是对我的下一个 Pet Clinic 还是坚持使用 Java。

为什么有些编程语言会死而有些能活下来?

谷歌打算要改变我们这个世界写软件的方法。近年来,这个搜索引擎巨头试图在这个星球中已经最广泛使用的编程语言上做改进,已经推出了 2 种新的编程语言。

谷歌打算要改变我们这个世界写软件的方法。近年来,这个搜索引擎巨头试图在这个星球中已经最广泛使用的编程语言上做改进,已经推出了 2 种新的编程语言。

通过一种叫 Go 的语言,谷歌试图拿它来替换年事已高的 C 或 C++ 语言,希望它能提供一种更加快捷的在数据中心里开发大型软件平台的方法。而通过一种叫 Dart 的语言,谷歌想用它来替代 JavaScript,改进我们开发运行在 Web 浏览器里的软件的方法。

但是,不管这些新的编程语言多么的具有吸引力,我们不得不问一句,它们需要多久才能真正的流行起来——如果能够的话。毕竟,新的编程语言不停的诞生。但只有很少一部分能被广泛的接受。

在普林斯顿大学和伯克利的加州大学,两位研究人员试图在为什么有些编程语言能走进它们的黄金时代而众多余下的却不能的原因上贡献出自己的智慧。在一个他们自称为“业余研究”里,Leo Meyerovich 和 Ari Rabkin 调查了数万个程序员,梳理了流行的代码库 SourceForge 上超过 30 万个项目——所有的这些努力都是为了能清楚为什么老的编程语言仍然处于霸权地位。

“为什么没有语言能够真正的超越 C 语言?”Rabkin 问道。自从 C 语言诞生以来的 35 年里,我们操作系统和软件设计都获得了巨大的飞跃,但是,虽说 C 语言中这段时间里有了加强,但也有很多新的非常成功的语言出现,可 C 语言仍然是开发语言中的中坚力量。

“为什么我们不能真正的超越 C 语言 ?”

— Ari Rabkin

部分的原因,他说,是因为语言的设计者并不都具有一个让这些语言实用化的目标。“学院派人的一个习惯就是喜欢去研究解决没有人真正遇到过的问题,”Rabkin 说。 Rabkin 最近刚刚获得了伯克利加州大学的计算机科学博士学位,现在在普林斯顿大学做博士后研究工作。

Rabkin 说,学院派的人经常想要开发出一种不同凡响的语言,但他们却从来不思考如何能让这种语言变的实用。在一些案例中,他们在一些最简单的事 情上都做的很失败,比如说为这种新语言写文档。在另一些案例中,设计者不停的往一种语言里添加新的的特征,成功的使试图使用这种语言的技术人员的大脑因超载而宕机。

“这样的问题的解决办法并非都是技术范畴,”Meyerovich 说。“我们需要去发明一些能够被“大众了解”的语言。

从30岁到35岁：为你的生命多积累一些厚度

在最近的一年的职业规划咨询过程中，我明显地感觉到 35 岁以上人群对于职业生涯规划需求的迫切性。也正是从这些案例中，我们得以清晰地洞察到，时间点的把握对于一个人的成长如何起着决定性的作用。

常常思考一个问题：是不是考虑做 2 年开发，打 2 年酱油，然后结婚生子，这样到底行不行？无论你是男是女，人生是一场独自修行的道路。如果在可以选的时候，还是选择靠自己吧。

无论你是要养家还是要实现人生价值，如果你处在迷茫之中，希望这篇文章可以传达一些正能量，可以帮到你。-- 当然这篇文章是转的呀，但真的很有理啊，假如看了这篇文章，会影响到你的一生，那么我就觉得今天这一小时的时间花在这里值了。

你所有不曾料想过的问题，都会随着时间的推移而与你不期而遇；你所有曾经潇洒的随遇而安，同样也会随时间流逝而让你承担那些似乎命中已经注定的代价。在这个世界上，“唯一不可阻挡的是时间，它像一把利刃，无声地切开了坚硬和柔软的一切，恒定地向前推进着，没有任何东西能够使它的行进产生丝毫颠簸，它却改变着一切。”我始终相信一句话：出来混，迟早要还的。虽然你我皆是凡人，只是这芸芸众生中的普通一员，但我依然希望每个人的生命都能够迎着太阳开花结果。不管以你现在的阅历是否能够理解这段话的涵义，请先记下来。我相信总有一天，你会明白。因为，时间能解释一切，时间能证明一切，时间能解决一切。——题记

在最近的一年的职业规划咨询过程中，我明显地感觉到 35 岁以上人群对于职业生涯规划需求的迫切性。也正是从这些案例中，我们得以清晰地洞察到，时间点的把握对于一个人的成长如何起着决定性的作用。在我们的客户中，遇到的往往是两类较为极端的案例：一类是已经做到一定级别——至少是总监级以上，在公司具有一定地位，年薪不少于 20 万的人；另一类是工作多年，但依然处于一个相对低的位置，无论是职位层级和物质回报，还是个人的价值感，均无法得到较高认同。这两类案例虽然极端，但却给我们提供可以借鉴的思考。通过对比，我们发现，但凡那些在职业发展上获得一定成功的人，都有一个共同的特征，那就是在自己所熟悉且擅长的领域，至少精耕细作了 10 年以上。而那些在职场上找不到自己位置的人，往往属于每隔一两年换方向，从来没有在某一个方向上深入积累下去。当然，除了频繁跳槽的因素之外，还有另外一个因素，那就是：已经在一个方向上深入积累，但这种积累属于重

从 30 岁到 35 岁 :为你的生命多积累一些厚度 II

复劳动式的,并没有上升,致使职业发展原地踏步。

不管你是否承认,你都必须重视“35 岁现象”。很多企业在招募人才时,明确规定年龄在 35 岁以下。如果你的年龄到了 35 岁却还在通过招聘网站投递简历不断跳槽的话,你就应该反省一下自己到底哪里做错了。当然,根据我们的实践咨询经验来看,如果你真到了 35 岁甚至更高的年龄才去思考这个问题的时候,很有可能这个问题你已经无力解决了,很多现实的困难会让你有心无力,束手无策。到了这个时候,很多人会因为当初的选择后悔不迭,但却欲哭无泪。所以,无论是为了避免走更多的弯路,还是迈向更大的成功,你都必须提前思考你未来的谋生之路。已经有太多的案例证明 :未雨绸缪会比临时抱佛脚有用得多。

在我们的客户中,有相当一部分属于 80 后,也是职业规划问题的高发区。年龄最大的一批 80 后,已经过了“三十而立”的年龄 ;但还有相当一部分 80 后,正在迈向三十而立的路上。如果说年轻是上帝给予你的犯错的资本,你还有时间去弥补 ;但如果你已经到了二十岁的尾巴上,这种资本将不再是你的专利。从大学毕业,到你的而立之年,这个时间已经足够长,让你有足够的机会去了解社会,适应社会,并反思自己的成长。

从 30 岁到 35 岁,这其中有 5 年的时间。假如给你足够犯错的时间,那么,为了不让你 35 岁以后的职业生涯变得一塌糊涂,你至少应该在 30 岁就确立明确的目标,并利用 5 年的时间去追赶。这可能是你成长的最后的最佳时机。错过了这个时机,你已不再年轻,社会也不会再以包容的心态去原谅你的年少轻狂。否则,你多走一步错路,就必定要在以后以十倍的代价补回来。从 30 岁到 35 岁,你应该学着为你芸芸众生般的生命,多积累一些厚度,以便让你下半生的职业生涯不要在“假如一切能够重来”的悔恨和遗憾中度过。

30 岁 :你必须要面对的三大问题

第一个问题,就是家庭与责任的问题。

不要以为自己还年轻。不要以为有些问题离自己还很远。无论是颓废还是忙碌,你的时间都在飞快地逝去,你感觉自己在加速变老。有些问题,不管你是否愿意去面对,但你的成长阶段决定了你必须要承担与年龄相匹配的责任。大多数的人,到了这个年龄段,都必须要考虑一下家庭问题。如果你已经成家立业,你必须清醒地意识到 :你的职业到底该如何发展,才能确保让你支撑整个家庭的负担? 这不仅仅包括你的爱人,还包括你的孩子及父母(尤其是双方都是独生子女的时候,你要承担的是双方四位老人的供养)。到了这个阶段,你的压力是成倍增长的,但如果你在职场上的成长无法实现倍增,甚至还在以某种形式进行着倒退,那么,你以后的职业发展就会面临更多的压力。而这种压力,更多的时候,会使你没有余力去谋求更好的发展。很多人在跳槽时,往往会有一个很大的顾虑 :如果我跳了,如果收入没有现在的高,那么,现在的工作,我还会轻易再动吗? 说得更不客气一点 :我还敢轻易再动吗? 我还有那个胆量与勇气吗? 所以,要想行动,就趁早,趁你还没有背上家庭的包袱的时候,轻装上阵,拼尽全力向前冲,这是你唯一的选择。千万不

从 30 岁到 35 岁 :为你的生命多积累一些厚度 III

要在这个时候享受安逸,否则,你的后半生都将永远在碌碌无为中“被安逸”下去。

第二个问题,是能力与年龄的匹配度问题。

为什么很多企业在招聘人才的时候,明确规定要 5 年经验、10 年经验等等类似的工作经验要求呢?因为工作时间意味着与之匹配的能力等级。同样的工作,5 年经验和 10 年经验所积累的能力是不一样的,所能承担的责任也是不一样的。你的工作年限越长,往往也意味着你的能力越强,这二者之间是一种正向倍增的关系。但如果你违背了这种关系,那么,你就无法获得用人单位的认可,进而丧失更好的职业发展机会。在我们的很多客户中,其中较为棘手的一种情况,就是能力与年龄不匹配的问题。很多工作 3 年、5 年的人,甚至和工作一年的人在能力方面并没有什么太大的差别,所以当他们的职业想向上突破时,会遇到很多的阻力。导致这种情况的产生有两种原因 :一是频繁跳槽,没有在一个方向上积累,无一技之长 ;二是虽然在一个方向上长期积累,但只有第一年是成长的,剩下的几年都是在做重复劳动,原地踏步。

所以,对于那些年龄 30 岁的朋友来说,从现在开始,你必须要慎重审视一个问题 :从毕业到现在,我工作几年了?我身上所具备的能力是否与我的年龄相匹配?如果不匹配,那么,你一定要及早树立危机感,并跑步前进,以弥补与那些先知先觉者之间的差距,确保自己不会在竞争的过程中被他人挤下马。

第三个问题,是知识结构的构建与提升问题。

我们曾经服务过一个客户,做销售做了五六年,也积累了相当丰富的实战操作经验。有一次跳槽去应聘某知名快消企业的区域营销经理的职位。在面试的过程中,所有关于具体操作层面的问题,他都能够对答如流,但上升到系统层面及战略层面的问题时,他的脑子就一片空白。我相信在职场上做过五六年的人都有这种感受 :感觉在具体的操作层面,无论是流程,还是技巧与方法,都能够熟练掌握 ;但如果从更高一层的角度去看待问题,往往又不知道如何下手。

身在职场,不同职位等级的人,所做的事情是不一样的,他们所具备的眼光与思维模式同样也有差别。一个最基础的业务员,想着如何维护好终端,这是战术层面的东西,也是他的职责所在 ;但如果一个营销总监这样的角色,还和业务员一样天天想着如何去和终端老板打交道,那就是他的失职了。每一个企业的运营,都会由战略与战术所构成。大的战略会细分成小的战略,小的战略会细分成一个一个的执行战术,由基层人员去付诸实施。不同层级的人,分别负责不同高度的工作,各司其职,这也是团队协作的意义所在。同样,每一个不同层级的人,也存在不同的知识结构。层级越高,你看问题的眼光和思路就要越高,你的整体知识结构层次也要向上发展和突破。要不然,你的能力,永远只能停留在具体的基层操作层面上,不会有大的发展,职位上更不可能有上升。

另外,从沟通的层面来说,你的职位越高■

本文未完,请阅读原文 :
http://developer.51cto.com/art/201212/370429.htm

编者按

长期以来，“软件业”一直被视为“智力密集”型的“朝阳”产业，大多数从业者都受过高等教育，其平均素质居于社会各行业的前列，这个产业的顶尖人物被公众视为“知识英雄”，比如微软公司的创始人比尔盖茨……

金旭亮：软件天才与技术民工

长期以来，“软件业”一直被视为“智力密集”型的“朝阳”产业，大多数从业者都受过高等教育，其平均素质居于社会各行业的前列，这个产业的顶尖人物被公众视为“知识英雄”，比如微软公司的创始人比尔盖茨雄据世界首富之位多年，更是为人“津津乐道”。

16 年前我下决心开始学习计算机技术的时候，对这个行业也充满了自豪感。

然而世事难料，在 2009 年高考刚刚结束之际，中国权威出版机构社会科学文献出版社于 2009 年 6 月 10 日发布首部《中国大学毕业生就业报告（2009）》，其中“计算机专业”荣登“就业最困难的十大专业”光荣榜。

大约从 2003 年开始，我在北京理工大学这所 211、985 重点大学开设 .NET 课程，这件事后来给某些人“炮轰”　说我要将重点大学学生培养成 IT 民工，也曾有技术牛人谆谆教诲年青学生想学软件，离金某人远些……

“程序员”何时变成了“IT 民工”？

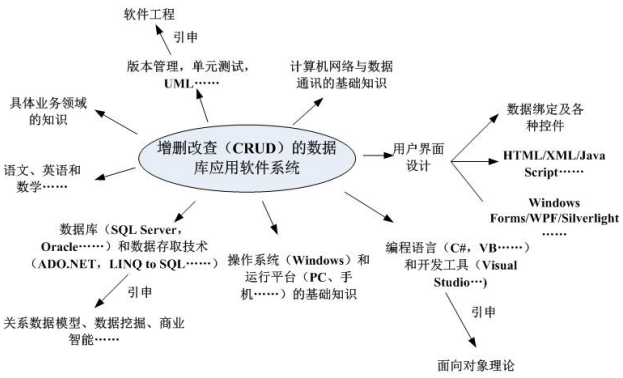
仅仅十多年，“程序员”就从“精英”沦落为“民工”，这也未免太戏剧性了吧！

“程序员”真的是“IT 民工”？

在中国，不少程序员都是“CRUD”程序员，这不能怪他们，因为中国多数项目都是 CRUD 的。所以，CRUD 似乎成为了“IT 民工”身上的标签，问题是 CRUD 这活真的象在建筑工地上倒水泥一样，不需要太高的智商和较多的训练？

下面分析一下需要从事这个 CRUD 的工作的程序员，到底需要哪些知识和技能。

我大致地思索了一下，画了以下这个图，展示出一名“.NET 程序员民工”要顺利参与一个 CRUD 的数据库应用项目开发所大致需要掌握的知识与技能：



说句实话，这个图虽然“挂一漏万”，但列出来的知识点与需要掌握的技能之多已经很“恐怖”了，对于一个需要掌握这么多知识与技术才能胜任与干好的工作，非要认为它是一个“低级”的“民工”就可以干的活，想必持这种观点的人一定知识丰富如汪洋大海，属“白骨精（白领－骨干－精英）”一族。

需要掌握这么多的知识，才有资格当一名合格的“CRUD”程序员，“IT 民工”的“门槛”其实挺高的，不是吗？

金旭亮 :软件天才与技术民工 II

事实也如此,软件开发行业从业人员的平均学历我估计至少是大专,很可能是“本科”,其中固然有一些只有“高中”甚至更低学历的“奇才”,但这仅是少数,而且这些人,虽然学历低,但绝不会是一个低智商的人。这说明软件开发行业从业者的“平均智商”应该是挺高的。

我在旅美学者薛涌 写的《天才是训练出来的》一书中找到了美国的一些统计数据 :

美国大学本科生的中等智商为 118,学士学位获得者的中等智商为 123,优秀毕业生的中等智商为 133,物理学博士的平均智商为 140。而在一般的人口中,中等智商为 100。也就是说,你的智商如低于 100 分,你就属于偏笨的人,高于此线,就属于偏聪明的了。

老外依据数据得出结论 **:要把书读懂读好,智商还真的得达到一定的水准** 。所以,能上大学并且顺利拿到学位的人,基本上是属于比较聪明的那一批人。

我们中国也有类似的观点,比如经常可以听到某家长叹息其小孩子成绩不好 :我家的那个小祖宗,看来根本就不是读书的那块料! 套用老外的观点,就是说这个小孩可能智商达不到上大学的基本要求。

老外什么都喜欢数字化,所以动不动就来个什么“智商”测试,而中国虽然不讲智商这一套,但其 实高考就可以看成是一次智商测试。在 80、90 年代,那时的大学生,经常都是百里挑一、几十里挑一,记得 89 年我参加高考,那年我们省的录取率还是十几取 一。所以那个时代的大学生被称为“天之娇子”,并非“空穴来风”。

重大的变化发生于 1999 年,从这一年开始中国大学开始扩招,尤其是计算机专业,更是扩招的重点,其规模迅速膨胀,录取率大大地提高,其结果是,用一句打趣的话说,“阿猫阿狗都去读大学了”,中国大学没有做智商测试的制度,不然,我想应该会看到 21 世纪在校大学生的平均智商比上世纪 80、90 年代的在校大学生的平均智商要低。但扩招对一些历史悠久声誉良好的重点大学的影响就没那么大,因为不管你 怎么扩招,能进名校(比如北大清华)的学生,智商是不会低的,因为其入学竞争仍然十分激烈,各方面差一点的根本就没这个机会。

老外还给出了一些著名科学家的智商 **:牛顿 190,伽利略 195,开普勒 175,达尔文 165,哥白尼 160**。

我对这些数据抱有怀疑 :牛顿和伽利略等的那个时代有智商测试吗? 这些数据是从哪计算出来的?

但这些著名科学的智商高于普通人,我想不会有人怀疑。

现在回到软件开发这个行业,其从业者平均学历为大专以上,应该说明这个行业需要的是整个人群中比较聪明的那一批。

这个观点与我的实际感受是一致的。

我在 IT 业“混”了十多年,深感这个行业牛人辈出,别说这个行业的“顶尖”级别人物,就是在 许多普通的 IT 企业、科研院所中,也有不少的让人佩服的牛人。我在北理工教了 8 年书,“笨”的学生实在没见着,而不少学生的聪明程度和学习能力等,都比我 在他们这个年纪时要强得多,

金旭亮 :软件天才与技术民工 III

还有一些学生,真的很聪明,悟性好,能举一反三,在技术上没多久就可以跑到我前面去了。

所以,软件开发应该是一个需要有较高智商的行业,我觉得不会有太大疑义。

为什么智商重要? 我在薛涌 写的《天才是训练出来的》找到了以下描述 :

在美国的白人中,生于占人口 5% 的最低层的穷孩子,日后生活在贫困线以下的可能比那些生于占 人口 5% 的最富裕阶层的孩子要高 8 倍。但是,那些智商属于最低的 5% 的孩子,则比那些智商属于最高的 5% 的孩子日后生活于贫困线以下的可能性高 15 倍。也 就是说,你的脑子是否好使,比你的家庭是否富裕对你的未来重要得多。

中国人有句俗话 :三百六十行,行行出状元。这些 "状元" 的出身并不重要,但他们应该接近 100% 的都有高的智商。

智商这个东西非常奇特,老外研究了这么多年,也没研究明白很多关键问题,比如智商是不是纯天生的? 后天不能提高? 如果能,那能提高多少? 如何解释一些高智商的人一辈子成就平平? 如何解释某些天才在某个领域内出类拔萃,在其他领域却连普通人都比不上? 还有,更重要的一点 :

普通人付出艰苦的努力,能成为天才吗?

我不是心理学家,下面我仅就想针对软件开发来聊聊自己对这个问题的看法。

你能成为软件天才吗?

在整个计算机领域,有一些老外的名字如雷贯耳 :冯诺依曼,图灵,比尔盖茨、Anders Hejlsberg,在中国,诸如 "求伯君"、"柳传志"、"李开复" 等名人的故事也是众人皆知,虽然我们拿不到其智商测试的分数,但相信他们个个都是聪明人。

"高智商" 其实是成为 "天才" 的必要条件。相应的另一个结论就是 :普通人再怎么努力,也成不了天才。

这话看上去令人泄气,但这是客观事实。

天才在各自领域内所达到的高度,大多数普通人是永远达不到的,比如如果有某人告诉我 :你只要努力,就可以比 Anders Hejlsberg 还牛。我一定会认为他在开一个让我很生气的玩笑。

回想起当年的高考,我费了九牛二虎之力,总分才刚够重点大学的最低分数线,后来的考研,我要 考 3 次才侥幸过关,我也曾参过北大的博士生入学考试,结果惨败而回,以后知趣地不再尝试……,我就这智商,普通人一个,所以您就别忽悠我了,我才不信天天 坚持跑步,日后就可以参加奥运会的马拉松比赛呢!

个体间智商的差异是客观存在的。

但成为不了天才不能成为躺倒不干的理由。因为这里有一个引发了巨大争论的问题……

智商重要还是努力重要?

国外心理学的统计数据明确无误地说明了智商的重要性,但这里有一个问题 :就算是具有 "天才" 浅质的人在总人员的比例很少,但人类有数十亿之多,"天才" 应该不少啊? 怎么数千年来,公认 "天才" 的就是牛顿等那几个?

1973 年,诺贝尔经济学奖得言、人工智能■

本文未完,请阅读原文 :

http://developer.51cto.com/art/201211/366922.htm

调查：2013年十大急需的热门IT人

作者 :文良

9 月 25 日消息，据国外媒体报道，著名 IT 杂志《计算机世界》日前针对各大公司计划招聘 IT 专家而发起的一项预期调查显示，在 334 名 IT 界高管中，有 33% 的人称将在接下来的 12 个月中计划增设更多的 IT 业领头职位。

当然，高管们也表示他们更需要的是能将现有技术转变成能在商界中竞争的有力产品。下面就是该调查结果总结的 2013 年急需的十大热门技能。

1、程序和应用开发技能

调查显示，有 60% 的 IT 高管们计划将在接下来的 12 个月招聘这样的人才。据资深人士里德（Reed）称 :“对公司来说，技术和软件是提高生产力、降低成本以及创建更好网页形象的最佳利器。”他还补充说，IT 公司将需要能创造并提高这方面技术的员工。

2、项目管理技能

调查显示，该技能的需求获得了 40% 高管们的认可。据美国放贷公司 Quicken Loans 软件工程副总裁杰米 – 汉米尔顿（Jamie Hamilton）称，对项目管理职位的需求部分是因为应用与应用之间越来越紧密的联系增强了现有项目的复杂性。

3、咨询台 / 技术支持技能

据纽约蒙特法沃医疗中心（montefiore medical center）的副主席杰克 – 沃夫（Jack Wolf）称，他正寻求不仅会建立和使用系统而且还会给予其他员工技术支持的新员工，他说 :“新的系统意味着你必须有更多的咨询台来处理更多的咨询量。”

4、安全技能

调查中，有 27% 名高管表示他们需要这项技能。安全一直是 IT 界高官们所关心的问题，为保护其系统以及越来越复杂的数据库，他们对安全专家的需求与日俱增。

5、商业智能 / 逻辑分析技能

该调查中，有 26% 的人认为这项技能很重要。最佳的候选人必须具备技术知识，商业知识以及强大的数据和数学分析背景。这项技能并不普通，一些公司目前正在招聘统计学家并教授他们有关技术和商业的知识。

6、云技术 / 软件即服务技能

据调查，有 25% 的人士目前急需这项技能。美国富乐公司此前表示，随着富乐拓展了两家数据中心，该公司目前需要云计算专家来帮助他们有适当的支持技术。

7、虚拟化技能

调查发现，有 24% 的高管表示将招聘这类相关人才。银行高管乔恩 – 比斯克纳（Jon A Biskner）表示他想在该行设立虚拟化管理一职，但目前这样的人才很难被发现，虽然 IT 专家们谈论虚级化技术，但在这方面有丰富经验的人则很少。

8、网络化技能

调查中有 19% 名高管计划要招聘此类■

本文未完，请阅读原文 :

http://developer.51cto.com/art/201212/369661.htm

从程序员到项目经理：不要死于直率

直率听上去是一种美好的品德，然而如果不注意区分实际情况，直率可能会成为一把伤人害己的“双刃剑”！下面我来来详细的了解关于直率的详细内容……

直率听上去是一种美好的品德，然而如果不注意区分实际情况，直率可能会成为一把伤人害己的“双刃剑”！

1. 直率是关于说话的问题

公司曾有一位人力资源经理是从传统行业转过来的，有一次她跟我说：“程序员真有意思，他们全都是一根肠子通到底，大脑不会转弯！”

还真是这样的，估计没有哪个行业的人员像程序员这样，具有同样的鲜明的性格特征：直率。

直率很容易理解，其实就是一个关于说话的问题，准确的说这是一个关于说还是不说、以及说多少的问题。 过于直率的人，在说话方面往往有两个特征：

(1) 想到什么说什么

这是一个说还是不说的问题。显然，不是什么东西都可以随时随地的说，或者对任何人说。可是对于过于直率的人而言，想到了就要说出来，就像俗话所说的：“嘴上没有把门的”，不管好话坏话，不区分场合，不论说话对象是谁，心里想着什么，嘴里马上就出来了。如果是好听话还好，皆大欢喜；如果是让人难堪的话，那就会伤害了别人了。如果是在公众场合让人难堪，人家可能会记住你一辈子。

(2) 知无不言、言无不尽

这是一个说多少的问题。“知无不言、言无不尽”毛主席老人家曾大力倡导的，这在讨论具体事情时无疑大有帮助，可是如果作为一种品格，在中国国情（文化氛围）下，还是不宜提倡。话并不是说得越多越好，说得越多，错的机会就会越多，反倒容易被人家抓住弱点或把柄。俗话说：“逢人只说三分话”，对一个项目经理而言，更应是如此。

在情绪方面，过于直率的人，往往不能很好的掌控自己，在说话时容易显得消极或冲动。

(1) 消极型

言语显示出消极的态度，例如给人泼冷水。给项目组成员安排工作时，我最怕听到两种话，一种是说“这个我不干了”，还有就是说“ 我不想干这个”，每次听到这两句话，心里就觉得凉飕飕的，虽然不爽，但作为团队的核心，我只能选择耐心的分析和开导。如果项目经理也以消极的方式来应对，整个团队的士气将会爱到打击。

(2) 冲动型

虽然人人知道“冲动是魔鬼”，但魔鬼不是那么容易掌控的，人难免有时会说出冲动的话来，以直率著称的程序员更是如此。冲动的主要表现是言辞激烈固执、情绪激动，一旦过头就会变成人身攻击。我经常看到程序员与项目经理讨论问题，后来争论不休，项目经理被迫使用自己的职位权力，强制执行，但这样可能会导致矛盾激化，搞不

从程序员到项目经理 :不要死于直率 II

好程序员就会拂袖而去 :“我不干了!”

无论是消极还是冲动,这都是职场的大忌。这些外在的情绪,在领导的眼中,就是体现出了一个人的工作态度,而在职场上,态度决定一切。对于领导而言,没有积极、合作的态度,就意味着你不能为我所用,那留你何用?!

2. 直率的悖论

对于直率的人有很多有意思的词语,好听一点的比如“直性子”、“直来直去”、“心直口快”、“快人快语”、“率性而为”,不好听的如“口无遮拦”、“大嘴巴”、“一根筋”、甚至“缺心眼”。可见在要不要直率这个问题上,中国人真的过得很矛盾、很痛苦,有时甚至无所适从。“说,还是不说,这是个问题”。长此以往,对于修养不够的人,造成人格分裂的倾向也不足为奇。

(1) 书上提倡直率,现实鼓励含蓄

其实好像没有哪本书正儿八经的告诉我们做人要直率,那为什么很多人眼里直率是一种美德呢? 我想这是从小到大的教科书对我们潜移默化的结果。书上教育我们世界上主要有两种人 :好人和坏人。好人多是直率的人,他们流芳青史,而坏人多是阴险狡诈之徒,他们遗臭万年。对比之下,我们当然想做直率的人了,光明正大,而且要直率太简单了,人人都做到,做的过程中还很爽。

另一方面,中国人的含蓄又是出了名的。中国是一个讲人情的社会,什么东西最重要? 人情和面子。一个成熟的人不会当面伤害别人的感情,不太好的事情喜欢用暗示,以免伤了面子。 不但说话,连中国的诗歌、中国的医学、中国人谈恋爱,也是含蓄的、有点模糊的不清的。

(2) 觉得直率好,却很少有人喜欢别人对自己直率

假如做一个调查,你是喜欢别人直率,还是含蓄,我相信大部分会选择直率。当别人含蓄的时候,我们会催着让他“有什么话直接说”,等别人说了,如果事情难办,心里又可能犯嘀咕,“这人说话也太直接了,连退一步的余地也没有了”。

其实我们很少有人希望别人每件事情都对自己直来直去,试想一下,当别人当面揭露我们的缺点时,我们会是什么样的心情? 与此相对照的是,我们又希望自己对别人直来直去的时候,别人会高兴的接纳,这可真是奇怪的事情。

(3) 网上“直率”,现实中彬彬有礼

有些人在网上非常的“直率”,直率到了一不高兴随时骂娘的地步,骂了又怎么样,反正见不着你。我相信在现实生活中,他们大部分都是讲道理有礼节之人,因为在现实中,我从来没有碰到像网上那么多人动不动就爆粗口、骂娘,祖宗十八代什么都可以骂。这样的人,有人格分裂倾向。一个人格完整健全的人,应该是一个不管什么场合都言行一致的人。

(4) 表面直率,心里打算盘

在中国什么类型电视剧最热? 我想无非是武侠剧、宫廷剧、历史剧,这些电视剧无一不充斥着人与人之间勾心斗角的故事。有一位伟人说 :“与人斗,其乐无穷”,试想一个什么都藏不住的人,与人斗,还不被别人玩死啊? 因此,无耻的政客们不论心里面打着多坏的算盘,表面上也得笑着,一副给人家交了老底了样子,好叫别人不要防着。当然,职场中还没有这么夸张,那是因为我们没有那大的利益需要去争斗,但中国人长期受这

从程序员到项目经理 :不要死于直率 III

种文化的熏陶,表里不一的人到处都是。

上面这些所谓的“悖论”只为了引起大家的思考,对于直率是好是坏这样的问题,并没有标准答案。但凡事过犹不及,过于直率无疑不但会伤害别人,最后也会反过来伤害到自己。

3. 直来直去伤人害己

直率这种性格听上去还不错,显得一个人光明正大,无所畏惧。有些人在自我介绍时候会说:“我这个人就是个直肠子”,言语之间还透着一些小小的得意。

很多人将直率视为一种美德。美德应当是对别人对自己都有好处的事情,可是“直率”并不是这样,如果把握不了度,可能反而会伤人害己而不自知。

(1) 伤人

要说直率伤人,相信人人都懂,很多人还会有切身的体会。国人最讲面子,一旦面子被伤,很难挽回,正所谓“刀伤易痊,舌伤难愈”。武侠小说中江湖中人,为什么整天打打杀杀的?直率就是一个重要的原因,那些人个个都是直率之人,又不讲礼节,经常出言不逊,一言不合,就兵刃相见。好在祖先教我们注意礼节,用礼节约束我们的言语和行为,社会就会和谐多了。

有一次我收到一个程序员的辞职申请,让我奇怪的是,这名程序员一向工作踏实,为什么突然辞职呢?跟他谈过之后,这才明白原因很简单,就是因为项目经理多次批评他“怎么这么简单的事情都做不好”,而他认为事情并不简单,是项目经理不了解实际情况。但他不想解释,因为他的自尊心被伤害了,不想再待在这里工作。我又找项目经理沟通,项目经理说这是无心之言,只怪现在的员工太脆弱。最后的结果就是员工离职了,项目进度也受到一定的影响。试想,如果项目经理在说话的时候更加注意措辞,怎么会有这样的结果呢?

(2) 害己

还记得《三国演义》中的杨修是怎么死的吗?杨修有过人之才,总是能看穿曹操的意图,然后得意洋洋的告诉别人,最后被曹操以扰乱军心的罪名处死。在历史上,许许多多的忠臣最后都没有好下场,因为他们往往是刚直之辈,言语伤了领导的面子,而领导一旦报复,后果是很可怕的。

直率就像没有成熟的柿子一样,好看不好吃。现实中,只听说过因为直率吃大亏,没有听说过谁因为直率升职加薪,不是这样的吗?一个管不住自己嘴巴的人,小则在团队中难以与人和谐相处,大则可能会得罪客户、甚至泄露公司机密,这样的人,领导怎么敢委以重任呢?

一个人能力很强,却因为说话的问题而吃亏,那实在不划算。技术的成长需要日积月累,而一句未经思考的话就有可能毁掉你在公司的前程。因此在把话说出去之前,我们应该多思考,三思而后言,避免犯下言语上的低级错误,后悔莫及。

4. 三思而后言

在计算机中,我们常用 IPO(输入 – 处理 – 输出)图来描述一个过程或一个功能模块的设计,其实人说话也是一个输入 – 处理 – 输出的过■

本文未完,请阅读原文:

http://developer.51cto.com/art/201212/371485.htm

淘宝王琤：Taobao JVM的性能优势与价值体现

北京时间 2012 年 12 月 4 日 Velocity China 2012 Web 性能与运维大会在北京召开，在会中 51CTO 记者有幸采访到了阿里集团的王琤老师，王琤老师主要负责淘宝 JVM 的开发工作，基于 OpenJDK VM 为淘宝定制、优化更加贴近应用需求的专用 JVM。

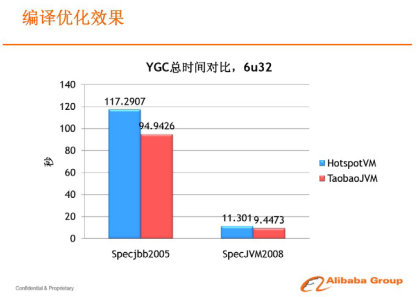
【51CTO 专访】北京时间 2012 年 12 月 4 日 Velocity China 2012 Web 性能与运维大会在北京召开，在会中 51CTO 记者有幸采访到了阿里集团的王琤老师，王琤老师主要负责淘宝 JVM 的开发工作，基于 OpenJDK VM 为淘宝定制、优化更加贴近应用需求的专用 JVM。而在此次 Velocity 大会上，王琤老师主要为大家分享了《淘宝 JVM 优化实践》。也借此机会，51CTO 编辑针对淘宝 JVM 这个话题跟王琤老师进行了一些沟通，分享给大家。

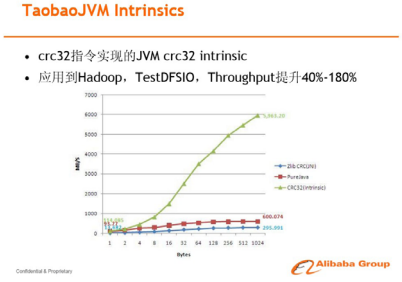
王琤（长仁）（右）正在与记者探讨

以下是采访内容：

51CTO：王老师您好，很多关心 JVM 技术的人都知道淘宝推出了开源的优化定制 JVM 版本：Taobao JVM，那么和 Oracle 官方的 JVM 版本相比，在性能上有哪些优势？

王琤（长仁）：一提到淘宝 JVM，大家肯定是关心性能，，性能是大家直观感受到的不同。从性能上来说由于 Taobao JVM 经过优化定制和官方版本相比在性能上有所提升。对于具体应用来说，根据很多应用特殊的性能需求我们有相应的优化点，比如刚才讲的我们有很多的 Intrinsic，这是在 oracle 官方版本里不可能做的，因为这些需求不是很通用。对于 hotspotvm 它更关注更加通用化的性能改进。而淘宝 JVM 更贴近淘宝的 Java 应用，我们可以接触到一线的性能需求，根据这些需求进行优化。所以淘宝 jvm 的性能提升的幅度在刚才 PPT 里面大家可以看到，会很有优势。这不是说我们做的比 Oracle 好，而是说我们更贴近于用户需求，我们按特定需求做一些特定的优化 . 一定会比通用的优化更好，这点是我们最有优势。





2011 :那些逝去的 IT 英才

51CTO :在这个开源项目上,您主要负责哪些工作?

王珵(长仁):我主要是负责专用计算组,我们开始做 taobao JVM 的时候只有我一个人,那是我刚刚来到淘宝的时候开始。大家有需求说我们淘宝以前对 JVM 的工作其实只能停留在参数调优,深入的去改它之前没有做过。我是因为来自 Oracle,对于这方面接触过一些,所以我们就成立了这个组来专门的做 taobao JVM,现在我们组的一部分同学在做 taobao JVM,另外一部分在做专用计算的工作。

对于淘宝 jvm 这个项目,我的老板 --- 章文嵩博士最早提出来要做这块,并且给予了很大的支持,他算是最初发起人。这个工作是淘宝需要的,因为淘宝是 Java 技术的最大应用方,是非常需要自己定制的 JVM 的。

51CTO :从 JVM 优化、定制及相关工具开发,有没有遇到哪些问题?您是如何去解决这些问题的?能否给网友分享下?

王珵(长仁):其实遇到的问题非常多,最大的问题还是人的问题。因为刚才讲到 JVM 的社区是非常小的,这部分的工作没有人很熟悉它,也因为社区太小,做这个东西的人很少。因为这点,所以相关的人才很难找到。这个最大的问题也就解释了为什么我们从一开始做到成果大规模应用经历了近 2 年的时间,在这么长的时间里主要的工作其实就是在培养人才。我们组的绝大部分同学其实都是应届的毕业生,从应届毕业生就开始培养,经过近 3 年培养,他们在这个领域已经做的很不错了。所以最大的问题还是在人的问题,而不在于技术本身。人才的培养我觉得环境很重要,像我们阿里集团的核心系统部,这个环境就是一个非常好的环境。可能对于一位应届生来说,他可能刚来的时候是一张白纸,当然在这么一个技术氛围非常浓重的环境里他可以快速成长,只要他自己努力就会成长的非常好。的团队,。来到这里的应届毕业生在 2-3 年的锻炼能达到这个水平已经是非常好的了。有的同学经过他自己的努力、兴趣并在我们这个环境里不断磨练,已经跳到美国 OracleJVM 团队参与 JVM 核心的开发工作。昨天正明(章文嵩博士)讲到我们在向 Oracle 输出人才,这一方面肯定了我们的工作,当然对于我们来说其实是好事也不是好事。所以环境是比较重要的,就是说你在这个环境里头会非常快速的成长。当然也借助这个机会希望更多有兴趣的同学能够加入到我们这个环境来做更多的事情。

51CTO :从淘宝,天猫全部替换了 Oracle 官方的 JVM 版本后,您觉得淘宝 JVM 的价值主要体现在哪方面?

王珵(长仁):线上很多已经替换了 Oracle 官方版本,价值主要体现在对应用方从心理上来说他们更加放心了。因为以前的时候 JVM 出问题,很难得到 Oracle 的支持。我们在 Oraclebugdb 上提 Bug 的时候反馈很不及时。现在用了我们的 jvm 话,我们组就会帮助他们解决问题,并且会专门为他们的需求来做一些 JVM 相关的工作,他们就觉得非常的放心。我们可能不用刻意关注具体的,比如性能,指标,功能点这些,单单从这一点来说其实大家就愿意来用我们的东西,因为更贴近他们,而且我们的服务很好。对于应用来说我们基本上都是面对面的上门服务。

51CTO :是否开放共享? ■

本文未完,请阅读原文 :

http://developer.51cto.com/art/201212/371819.htm

大数据成为捕捉网络威胁的眼睛

数据爆炸式增长已经成为我们无法回避的现实，而大数据分析技术则应当成为我们迎接挑战的有利武器。

【51CTO 外电头条】“数据已经成为新时代的‘石油’资源。”

最近几个月大数据的激增人气再次显露出来。对于我们这帮搞技术的人士而言，数据始终是贯穿生活始终的核心价值所在。但在过去数年中，它的价值则变得更加透明。无论是智能手机、基础设施所在地的电网体系还是硅谷新兴企业的运作模式，都开始围绕“数据”这一概念大做文章。人民群众纷纷表示，数据代表着新一轮潜在的企业盈利能力。

（当然，电力仍然是一切的保障。试想我们坐在没电的办公室里，盯着手机上即将归零的电池剩余，数据什么的自然只能是浮云。但这好像跟今天的话题没啥关系。）

几十年来，数据已经成为企业运营的润滑剂与推动力。如今，“大数据”这一涉及庞大复杂数据合集计算、整理与分析的模糊概念则带来新的价值增长点，并承诺为企业加速信息向财富的转化过程。

原因在于：随着数据量的爆炸式增长，对其加以利用的可能性也将相应提高。数据自身与生俱来的关联特性既成为前所未有的发展机遇，也带来了诸多极为罕见的技术阻碍。

只要略加分析大家就会发现，发掘数据潜能的最大挑战在于选择理想的大数据解决方案。换言之，我们需要利用大数据来实现大数据保护工作。这跟电影《盗梦空间》的剧情设定如出一辙——利用梦境来改变梦境。

Seculert 公司是一家以色列新兴安全企业，其主要发展方向是利用大数据分析技术捕捉企业中的网络威胁。今年十月下旬，他们推出了一款名为“Seculert Sense”的专有引擎，尝试利用 Amazon Elastic MapReduce 从活动僵尸网络、恶意软件及日志文件等由客户上传到云端的信息中收集数以 TB 的数据并加以分析。而分析结果将被传输到一款基于 Web 的安全专用控制面板当中。

研发初衷：加快威胁检测速度、提高防御体系灵活性，使现有安全措施更好地适应不断变化的网络威胁并解决日益壮大的外部网络企业活动所带来的安全难题。

为了了解更多信息，我对该公司创始人 Aviv Raff（简称 AR）与 Dudi Matot（简称 DM）进行了一次专访。

我：Seculert 公司是如何建立起来的？

DM：这家公司于 2010 年诞生，但我们在那之前就已经开始关注这块市场了。

大数据成为捕捉网络威胁的眼睛 II

如果把目光投向 2000 年之初，也就是 03 到 04 年之间，那时候恶意软件所针对的还主要是金融类消费者。而 2006 年左右依托僵尸网络实现的信息扫描与拒绝服务活动则成为主流。到了 09、10 年，攻击者的目标又开始针对企业环境。当时谷歌公司是第一家明确表示自己受到攻击的企业，据说当时的攻击活动来自中国。其后反应遭遇侵袭的公司逐渐增加，最终达到 70 多家，大部分安全服务供应商都被牵涉其中。

多数传统安全供应商只向客户提供管理政策或者基于签名的解决方案。他们研发相关工具，而客户则负责配置工作。其它的就不用指望了，方案已经成形，大家只能祈祷这些产品真有拒敌于千里之外的本事。

现在黑客们在入侵企业并获取专有信息方面的技术水平已经越来越高，因此我们一直在努力收集资料，以揪出那些无法被现有系统所发现的恶意活动。

目前大多数遭遇侵袭的企业尚处于懵懂之中，他们没有意识到自己的安全体系已经被攻破。就像一群目光短浅的小羊羔，他们躺在旧技术与旧概念打造的残舍破窑中，还以为自己非常安全。

说到这里，**我要感谢以大数据为代表的云计算及其它现代技术。在它们的帮助下，我们才有能力获取以 TB 计算的海量数据、以分布式方式通过精心开发的代码对其进行处理与关联，并最终拿出复杂而准确的分析结论。**

我 :您的公司非常重视企业网络之外的安全保护工作。我想自带设备趋势正是成就这一设想的重要因素。

AR: 现在企业员工会以远程方式接入内部网络——从家中、在旅途上、使用自行购买的移动设备等等。这种状况对传统管理方案提出了严峻挑战——过去的工具只允许企业管理属于自己的系统和设备。我们的产品则能够检测到来自外部的、存在安全问题的接入设备。

这套产品不会给企业管理者带来硬件或存储等方面的压力。对管理范围进行扩展其实非常简单——大家可以通过配置直接将日志记录范围从一个月提高到一年，整个调整过程一小时内就能完成。

DM: 许多应用程序及其它技术资产正向云环境转移。越来越多的员工开始以远程及外部方式处理内部事务。事实上继续在网络安全工作中苦苦挣扎已经没有任何好处——针对有限的几种设备进行复杂精密的检测，这实在有点吃力不讨好。相比之下，云环境就要友善得多。由于设备都在服务供应商那边，大家完全不必费心进行管理及维护。

大多数供应商所提供的设备都能带来良好的企业网络覆盖效果——通过网关或者其它机制。Check Point、Palo Alto 网络公司等都是不错的选择。但还没有哪家供应商能切实保护远程员工，目前大家只能指望这些员工自发安装杀毒软件并严格遵守安全政策。IT 部门正在行动，但他们在安全性方面还没什么实质性进展。目前流行的管理体系相对过去而言实在太过松散。

AR: 我们正努力帮助他们勘破这个复杂多变的时代、了解组织之外的真实世界，并把握新型恶意软件对现有安全机制的影响。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201212/370172.htm

航旅纵横-基于民航的数据整合之路

在 2012 年 12 月 4 日 Velocity 大会上，51CTO 专访了来自航空行业的唐老师。唐老师在本次大会上给大家带来的是跟海量民航异构信息处理的技术分享。

航旅纵横技术负责人唐先生本次分享中谈到了航旅纵横海量异构数据整合的问题，这些异构数据的来源有国内外民航系统的差异、也有部分是各地机场 IT 系统水平参差不齐，即使民航多年来 IT 系统的不断提升，但人员技能方面的差异、基础网络的稳定性等还是会造成异构数据问题。在这个问题上，民航等传统行业与信息行业有很大的不同。信息行业的技术革新一般都是颠覆性的，而传统行业大多都是打补丁的形式，必须对原有技术系统进行继承。这一点上，会对整体信息系统的效率造成一定的影响。

专访现场（51CTO 摄）

航旅纵横为旅客提供便捷的航空服务，其在数据整合过程中，会有一个数据信息取舍的问题。如何整理一份及时准确的数据，需要对源数据的字段进行有效识别，某些字段需要保留，某些字段必须舍弃。在这个平衡度的问题上，团队采用的方法是根据历史数据和流程，确定需要保留的字段。这是一种目前在用的思路。

不断增加的数据量级

中国民航业在不断发展，航线信息和旅客量在成倍增加。这些都会成为一个海量数据的来源，并且随着航空服务形式的多样性和社交媒体的爆发，航空数据不可避免的加入越来越多非结构化的数据。

未来民航业将会是一个大数据集中的行业，该如何来处理这些数据呢？

首先数据涉及很多敏感数据，需要进行高层次的保护措施以免泄露，保护信息的安全。同时对数据进行分析和挖掘，为旅客和企业提供相应的信息服务。对于数据的融合，采用系统隔离，规则的筛选和清洗，数据出口的数据校验等方式，形成了较为自动的数据模型。

对于 IT 技术人员，在设计系统架构时，还应考虑极端情况（比如恶劣天气等）系统的影响。唐老师表示自己的团队会考虑在现有系统的处理能力基础上，做出 2 倍到 3 倍的性能冗余。如果这些冗余都无法保证正常的运行时，会保证绝大部分用户的正常使用，而会把一些不重要的用户流量进行暂停、这些可以保证系统的安全和正常运行。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201212/371591.htm

MariaDB成为MySQL命运转折点？

MariaDB 项目带来开放与创新之希望，陷入困境的 MySQL 社区能否借此焕发生机？Oracle 的闭源阴谋能否得逞？

【51CTO 外电头条】当初 Sun Microsystems 公司即将迎来收购之时，一群曾经参与过著名人气开源数据库 MySQL 开发的程序员们决定另起炉灶，打造名为 MariaDB 的新项目。

新项目由 Michael “Monty” Widenius 定名并领导，这位 MySQL 项目的原始开发者兼 MySQL 公司联合创始人放弃甲骨文的招揽，从零开始重新奋斗。在离开 Sun 公司之后，他在自己的故乡荷兰成立了一家公司——也就是 Monty Program AB——借以管理 MariaDB 项目的开发工作，同时向广大 MySQL 技术达人敞开怀抱。不久之后，Monty Progam 公司就拥有了一个实力强劲的开发团队。

也许大家并不了解，但他们的确一直在废寝忘食地工作。由于甲骨文公司在全面收购 Sun 资产后对 MySQL 的开发工作表现出极高热情，导致 MariaDB 感受到了前所未有的竞争压力。然而优秀的人才在对抗当中力挽狂澜，帮助 MariaDB 站稳了脚跟。在一份由 Network World 网站公布的六大开源数据库评测报告中（包括 MySQL），我们发现 MariaDB 赫然成为人气最高的数据库方案。Monty 告诉我，MariaDB 与 MySQL 相比拥有约三十人工作年的研发优势，而 Monty Program 公司也对自家产品的领先性表示认同——尤其是在安全性修复方面。

快速修正是关键

Monty 告诉我们，MariaDB 开发团队一直在与 mitre.org 通力合作，希望保障一切安全问题都能被快速发展、上报且拥有完备的细节描述。由于甲骨文公司不再公布安全修复细节，MariaDB 团队通常需要对来自 MySQL 的补丁程序进行逆向工程，借以找出其修复对象。搞清状况之后，这些补丁将被合并起来并为 MariaDB 提供服务。Monty 表示 “MariaDB 可以被看作是安全性最高的 MySQL 版本” ——这一声明相当大胆。

尽管 MariaDB 与 MySQL 两者在新版本公布之前，其安全漏洞的修复细节一般都要受到严格保密，但最近的一次事件令双方在安全应对机制与响应速度上的差异显露无遗。问题甫一曝光，MariaDB 就行动起来并在几天之内就利用开放并记录开源补丁完成了修复工作；相比之下，MySQL 直接现在（截稿之日）仍然没能搞定这些安全漏洞。

这已经不算什么新鲜事了。甲骨文公司一直在对企业资源优先参与并处理 MySQL 社区事务的做法抱怨不已，结果当然显而易见：裁撤相关流程转投其它开发项目并延缓修复安全漏洞，而这一切都令项目组与 MySQL 生态系统间的交流日益恶化。我曾与 MySQL 与 MariaDB 双方的外部生态系统合作伙伴进行过对话，而他们都对

MariaDB 成为 MySQL 命运转折点？Ⅱ

甲骨文公司的强硬作风表示无奈。

面临如此窘境，上周传出的新闻无疑可算大大的利好消息：MariaDB 基金会正式成立，MySQL 开源社区也将有望自此步入新的发展阶段。MariaDB——从侧面来看也就是 MySQL——终于拥有了专门的机构体系，该基金会的出现将一举扭转发展控制一家掌握的被动现状。

生态系统受到影响

这对 MySQL 的生态系统又意味着什么？首先，MySQL 将借打包与整合之力获得提升。很明显，像 MariaDB 这样的开放式项目在使用便捷性上要远超过某家企业针对自身业务所打造的数据库方案。

在未来，我们很可能看到被囊括在 Linux 发行版中的 MariaDB，同样也可能在 LAMP 部署中发现它的身影（而且在用 MariaDB 代替传统的 MySQL 之后，我们仍然可以沿用‘M’这个字母）。

其次，创新的可能性也将大大增加。某位开发人士曾告诉我，MariaDB 身上具备一些极具吸引力的发展潜力——支持 OLTP、OLAP、以数据为中心的专业处理方案以及高度可扩展的多控制集群。

这种多元化的发展方向意味着项目需要接纳来自各个方面的信息与意见，而基金会的成立为开发者提供了一套透明的管理平台，其向所有人开放的基础特性对于整个项目的走向有着非常深远的积极意义。

第三，有效改善项目竞争力。MariaDB 最近刚刚公布了一套兼容性极高的客户端库，完全利用 LGPL 从零开始重新编写、借以替代 MySQL 及其衍生版本原先所使用的 GPL——现在 MariaDB 与 MySQL 双方都将由此而受益。正如社区成员 Arjen Lenz 在评论中所说，这一点对于双重许可问题意义重大。现在商业用户们再也不必为了避免额外的 GPL 合规性管理需求而忍痛为 MySQL 购买专有许可了。

将三个因素综合起来看，MariaDB 绝对有机会在规模庞大且对手众多的 MySQL 市场中依靠独特魅力取得竞争优势。Monty 告诉我们，MariaDB 项目的代码贡献者中已经出现了许多企业巨头的身影，其中包括 Facebook、谷歌、Twitter 等，而且专为开发者们准备的 IRC 交流平台上经常有上百人在线。

虽然形势见好，但也并非万事大吉。MySQL 生态系统中有那么一部分似乎就完全没受到正面影响：

Drizzle 项目，其目的是通过重新设计一套更小、更具模块化特性的微内核实现云部署。该项目创始人 Brian Aker 在 Twitter 上明确表达了自己对于基金会的不屑一顾，并通过电子邮件宣称自己将保持观望态度。

Drizzle 项目已经拥有了自己的一套非 GPL 客户端库——虽然还未能实现完全兼容——而 Aker 认为 MariaDB 所使用的 JDBC 驱动（即 Java 数据库连接）从某种程度上来说源自 Drizzle（其采用 BSD 许可，而 MariaDB 则采用 LGPL）。（本文未完，请阅读原文）■

http://database.51cto.com/art/201212/373195.htm

旅游会是开发者另一片需求蓝海?

再想靠切水果盈利已经慢慢不切实际,去开发一片新的需求蓝海迫在眉睫。中国老百姓的旅游需求应该被更多的开发者,特别是移动开发者所认识。

【51CTO 独家特稿】在天朝的微博上,流传着这样的传说“半个平米,你可以日韩或新马泰一游;一个平米,你可以游遍欧洲;半个卫生间,可以游遍非洲美洲;一个卫生间可以走遍全世界;等你游遍全世界,你的世界观也许就变了,房子已经不那么重要了。”

2011 年,我国旅游业三大市场继续保持较快增长。国内旅游人数达 26.4 亿人次,较 2010 年增长 25.5%;国内旅游收入 1.93 万亿元,较 2010 年增长 23.6%。入境旅游人数 1.35 亿人次,较 2010 年增长 1%;入境过夜旅游人数 5,758 万人次,较 2010 年增长 3.5%;旅游外汇收入 484.6 亿美元,较 2010 年增长 5.8%。出境旅游人数 6,900 万人次,较 2010 年增长 20%。全国旅游业总收入 2.25 万亿元,较 2010 年增长 20.8%。

周末的北京,呼呼的北风把笔者困在家中。笔记本电脑上的 QQ 一直不停的蹦出类似“泰国、沙滩、普吉岛、防晒霜”这样代表温暖的文字。QQ 另一端的同事正在不断的跟笔者讨论他们去泰国自助游的细节。内容从线路的选择,到机票的购买,酒店的预订无所不包。这是一种典型的 80 后自助游模式,它早已摒弃了传统的上车睡觉、下车看景的旅行团模式。现在的旅行者需要更多更加个性化、可定制的工具来帮助自己,这会是开发者们的未来目标吗?

交通,需求未被完全释放

在今年的 Velocity 大会上,笔者与来自中航信的一位技术负责人进行沟通了解到,中国的第一张飞机票就已经是计算机打印并存储的形式了,而近几十年的航空旅客数据都以磁带的形式存储在中航信的数据库里。可以说,中国民航业在很早之前就已经开始了信息化的过程。而在不远的将来,民航的信息将会更多的向公众开放,开发者将得到稳定的信息来源。

那将是未来的事情,在互联网和移动设备高度发达的当下,中国旅客依旧无法准确的掌握航班起降时间。当我们傻乎乎拉着行李箱到达机场,被告知航班取消的一刹那,你幸福吗?

尽管现在有“飞常准”这类航班信息应用。但它不能根据用户个人的身份证号,自动统一显示用户所有的航班情况。还需要采用用户手动输入订票信息和时间的方式,不能够让用户完全避免遗忘出行的功能。

另外一个交通出行大户——铁路,目前市场上能见到的多是列车时刻表和订票助手之类的应用。

用户除了能知道票价、开车时间之外,对于正晚点信息、座位是否靠窗等都很难查询到。而这些功能其实是可以实现的,至少被人诟病许久的 12306 上确实有正晚点信息的发布。(本文未完,请阅读原文)■

http://developer.51cto.com/art/201212/370313.htm

InnoDB数据库主从复制同步心得

近期将公司的 MySQL 架构升级了，由原先的一主多从换成了 DRBD+Heartbeat 双主多从，正好手上有一个电子商务网站新项目也要上线了，用的是 DRBD+Heartbeat 双主一从，由于此过程还是有别于以前的 MyISAM 引擎的，所以这里也将其心得归纳总结了一下。

1）MySQL 的 replication 过程是一个异步同步的过程，并非完全的主从同步，所以同步的过程中是有延迟的，如果做了读写分离的业务的话，建议也要监控此延迟时间；

2）MySQL 的 master 与 slave 机器记得 server-id 要保持不一致，如果一样的话，replication 过程中会出现如下报错：

Fatal error: The slave I/O thread stops because master and slavehave equal MySQL server ids; these ids must be different for replication to work(or the --replicate-same-server-id option must be used on slave but this doesnot always make sense; please check the manual before using it).

这个问题很好处理，即将 slave 机的 server-id 修改成跟 master 机器不一致即可。

3）我以前的一个误区就是，slave 机器是用自己的二进制日志来完成 replication 过程的，其实不是这样的，根据复制的工作原理 ;slave 服务器是 copy 主服务器的二进制日志到自己的中继日志，即 relay-log 日志(即 centos3-relay-bin.000002 这种名字的）中，然后再把更新应用用到自己的数据库上，所以 slave 机器是不需要开启二进制日志的，这样过程一样会成功的 ;除非是准备做主主架构，这才需要 slave 机器开启二进制日志，这个问题一直在导着我，我以一直以为 slave 机器搭建 replication 环境时是一定要开启二进制的。

4）在 master 机器上授权时，尽量只给某一个或某几个固定机器权限，让它们只有 replication slav，replication client 权限，尽量不要给 grant 权限；

另外，虽然数据库我们一般是通过内网操作，但越是在在内网对 MySQL 数据库进行授权操作，越是要注意安全；

5）replication 搭建过程按照正常流程走的话，一般很容易实施成功，如果出错的话，多检查下网络环境、权限问题，一般来说整个搭建过程应该还是会比较顺利的。

在数据库设计初期，我已经将此电子商务的数据库引擎定义为 InnoDB，除了数据库中原有的系统表之外，其它表全部由 MyISAM 转成了 InnoDB，原因有二：

1）电子商务业务会涉及到交易付款，在这种基本 OLTP 的应用中，InnoDB 应该作为核心应用表的首选存储引擎；

2）DRBD 系统重启时的过程会比较缓慢，会频繁的读表，如果表引擎为 MyISAM 的话极有可能出现损坏情况。

为了造成不必要的问题，我将数据库的表引擎由 MyISAM 均转成了 InnoDB 引擎的表。（本文未完，请阅读原文）■

http://database.51cto.com/art/201211/366891.htm

编者按

函数式语言和框架依赖运行时来控制日常编码细节诸如迭代遍历，并发执行和状态变迁，但这并不意味着你失去了对这些细节的控制权，在需要的时候你依然可以决定这些细节的行为。函数式思维的另一个要点是明确在什么时候想放弃多少控制权。

函数式思维：思维的功能（二）

在这个系列的第一期文章里，我先是讨论了函数式编程的一些特性，并用 Java 和函数式的语言展示了这些特性。在本篇文章里，我将继续讨论这些概念如第一级函数，优化器和闭包。但这期文章的主题是控制 :即什么时候用它，什么时候需要它和什么时候应该放弃它。

第一级函数和控制

列表 1 是上次使用 Function Java 库实现的数字归类器，它拥有 isFactor() 和 factorsOf() 方法 ：

列表 1 函数化的数字归类器

```
importfj.F;
importfj.data.List;
import static fj.data.List.range;
import static fj.function.Integers.add;
import static java.lang.Math.round;
import static java.lang.Math.sqrt;
public class FNumberClassifier {
publicbooleanisFactor(int number, int potential_factor) {
return number % potential_factor == 0;
    }
public List<Integer>factorsOf(final int number) {
return range(1, number+1).filter(new F<Integer, Boolean>() {
public Boolean f(final Integer i) {
return number % i == 0;
            }
        });
    }
public int sum(List<Integer> factors) {
returnfactors.foldLeft(fj.function.Integers.add, 0);
    }
publicbooleanisPerfect(int number) {
return sum(factorsOf(number)) - number == number;
    }
publicbooleanisAbundant(int number) {
return sum(factorsOf(number)) - number > number;
    }
publicbooleanisDeficiend(int number) {
return sum(factorsOf(number)) - number < number;
    }
}
```

函数式思维 :思维的功能(二)II

在 isFactor() 和 factorsOf() 方法里,我放弃了使用循环算法的做法 而把它变成了框架内的一个区间遍历。如果框架(或者说你选择了诸如 Clojure 或 Scala 这样的语言)已经优化了潜在的实现,那么你将自动获益。尽管起初你可能很不情愿的放弃这样的控制权,但是请注意这是编程语言和运行时的一个普遍发展趋势 :不久以后,开发人员将更加关注对细节的抽象,而平台会处理更多效率的问题。我从来没有在 JVM 上担心过内存管理的事情,因为平台允许我去忘记内存管理。当然,它有时会使问题变的更加麻烦,但是你每天所作的工作不就是为了在这方面作出一个较好的权衡么。函数式语言的设计诸如高阶函数和第一级函数允许我快速走到抽象的层面上,并且更多的关注在代码做了什么,而不是它怎么做。

正如 Functional Java 框架做的那样,用 Java 去写代码是那样的笨重,因为这个语言没有特意去设计它的语法。那么用函数式语言来写代码会是什么样子呢?

Clojure 的数字归类器

Clojure 是一个函数式的 Lisp 设计,它运行在 JVM 上。考虑使用 Clojure 来写一个数字归类器,如列表 2 所示 :

列表 2, Clojure 实现的数字归类器

```
(nsnealford.perfectnumbers)
(use '[clojure.contrib.import-static :only (import-static)])
(import-static java.lang.Mathsqrt)

(defn is-factor? [factor number]
  (= 0 (rem number factor)))

(defn factors [number]
  (set (for [n (range 1 (inc number)) :when (is-factor? n number)] n)))

(defn sum-factors [number]
  (reduce + (factors number)))
(defn perfect? [number]
  (= number (- (sum-factors number) number)))
(defn abundant? [number]
  (<number (- (sum-factors number) number)))
(defn deficient? [number]
  (>number (- (sum-factors number) number)))
```

列表 2 中的大多数代码都很容易理解,即使你不一个坚实的 Lisp 开发者 你也能够学习到由内而外的去读程序。例如, is-factor? 方法携带 2 个参数,并且当 number 乘上 factor 时会询问余数是否为 0。一旦你把代码和列表 1 中 Java 实现联系到一起后,类似 perfect? , Abundant? 和 deficient? 这些方法也应该很容易理解。

Sum-factors 方法使用了内建的 reduce 方法。Sum-factors 一次分解列表中的一个元素,使用函数作为每一个元素的第一个参数。Reduce 方法在这几种语言和框架里都有着不同的样子。你在列表 1 里面看到的就是 Functional Java 的版

本 – foldeLeft() 方法。 Factors 方法返回一系列数,因此我每次处理列表中的一个元素,并将每一个元素加到前面的累加和上,这个累加和就是 reduce 方法的返回值。你可以看到一旦你习惯于用高阶函数和第一级函数去思考后,那么你就可以过滤掉代码里大量的干扰因素。

Factors 方法看起来有点像是一个随机的符号集,但是一旦你理解了列表的解析过程(Clojure 中几个功能强大的列表操作功能之一),那么你就会觉得这么做是很有意义的。像前面说的那样,由内而外的理解 factors 方法是最容易的,所以不要被编程语言的术语所迷惑。For 关键字在 Clojure 中并不是指示 for 循环的意思。你可以把它当成所有过滤和转换结构的根源。这样我就可以要求它使用 is-factor? 断言(这个方法我在列表 2 里面已经定义过了 – 这是第一级函数的严格用法) 去过滤一组从 1 到 number+1 的数,然后返回匹配的结果。这个操作的结果是得到一堆符合过滤条件的数,这就是我强制一个集合去删除重复数的做法。

尽管学习一门新语言是一件痛苦的事情,但是当你理解了函数式语言的特性后,你会获得很多的解决问题的技能。

优化

切换到使用函数式样式的一个好处是你有能力去利用语言或框架提供的高阶函数。但是有多少次你不想放弃这样的控制? 在我较早的例子里,我拿迭代器的内部工作机制与内存管理的内部工作机制来举例 :大多数情况下你并不关心这些实现细节,但是在优化和进行类似调整的时候你必须关心它们。

我在 “思维的功能(第一部分)” 中已经展示了两个 Java 版本的数字归类器,我优化了代码里面的确定因子算法。原来的实现是基于取模操作的,这种做法效率较差,它需要从 2 开始去检查每一个数直到目标数本身以便确定这个数是否为一个整除因子。你可以通过成对的获取因子来优化这个算法。如果你想获取 28 的整除因子,那么一旦你找到 2 同时你也获得了 14。如果你可以成对的获取因子,那么你只需要检查这个数平方根以下的所有数即可。

这个优化在 java 的版本里实现起来很容易,但是在函数式的 java 版本里几乎是不可能的,因为我不能直接控制迭代器的实现机制。但是要学会函数式思考的一部分就需要放弃那种控制的概念,并发挥自己的另一方面能力。

我重申一下原来问题的功能 :过滤所有的因子从 1 到 number,保留所有能匹配 isFactor 断言的因子。这就是列表 3 所实现的 :

列表 3. isFactor 方法

```
    public List<Integer>factorsOf(final int number)
{
    return range(1, number+1).filter(new
F<Integer, Boolean>() {
    public Boolean f(final Integer i) {
    return number % i == 0;
        }
    });
    }
```

函数式思维 :思维的功能(二)Ⅳ

尽管从声明的角度来说列表 3 是优雅的,但是代码却有点低效,因为它要去检查每一个数字。一旦我理解了优化过程(以成对的方式捕获整除因子,并且只检查到目标数的平方根),我就能这样重申问题 :

1. 过滤目标数的所有整除因子,直到该数的平方根。

2. 用目标数除去每一个整除因子来获得对称的因子,并将它加入到因子列表中。

有了这个想法,我就能用 Functional Java 库来写出 factorsOf() 方法的优化版本。如例子 4 所示 :

例子 4。 优化后的 factors–finding 方法

```
public List<Integer>factorsOfOptimzied(final int
number) {
   List<Integer> factors =
range(1, (int) round(sqrt(number)+1))
.filter(new F<Integer, Boolean>() {
public Boolean f(final Integer i) {
return number % i == 0;
          }});
returnfactors.append(factors.map(new F<Integer,
Integer>() {
public Integer f(final Integer i) {
return number / i;
                              }}))
.nub();
}
```

列表 4 的代码是基于我前面描述的算法实现的,使用了一些 Functional Java 框架的时髦语法。首先,我取得了从 1 到目标数平方根加 1 的这个区间(确保我获得所有的因子)。其次,我将前一版本的取模操作封装在 Functional Java 的代码块中用于过滤结果。然后我将过滤后的列表保存到了 factors 变量中。第四步(由内而外的阅读),我拿到这个列表并执行 map ( )函数,这个函数执行我的代码块来处理每一个元素并生产新的列表(映射每一个元素到新值上)。这个列表包含了到目标数平方根的所有因子。我需要用目标数挨个的除于它们来获取对称的整除因子,这就是 map ( )方法干的事情。第五步,现在我拥有了一个对称因子列表,我把它追加到原来的列表里。最后一步,我必须统计这些因子的数量, 我的做法是使用列表来取代集合保存这些因子来帮助我统计。List 方法很容易进行这类操作,但是我的算法有一个副作用就是当平方根是一个整数的时候会出现重复数。例如,目标数是 16,那么它的平方根就是 4,这个时候 4 就会在整除因子列表里面出现两次。为了可以继续使用 List 方法,我只需要在方法的末尾调用 nub ( )方法就可以去除所有的重复。

由于你常常受迫于实现细节,那么即使使用高阶抽象如函数式编程,你依然可能让代码变糟。Java 平台大多数情况下在低层的构件上提供保护,但是如果你愿意,你依然可以降低到需要的层次。类似的,在函数式编程里,你通常愿意为抽象放弃细节,而去节省一些时间直到它真的成为问题。在所有的 Functional Java 代码里面可视化是占主导地位的,正如我已经展示过的块语法。它使用泛型和匿名内部类来作为一种伪代码块和闭

函数式思维 :思维的功能(二)V

包结构。闭包是函数式语言的一个有用的特性。那是什么让他们如此有用呢?

闭包有什么特别之处?

闭包是一个函数,它携带着一个隐式的绑定,这个绑定关联到它内部所有的引用变量。换句话说,函数或者方法封装着一个所有它引用对象的上下文。闭包在函数式语言和框架里常常被作为一个轻量的执行体,传送给高阶函数,如 map ( )。Functional Java 使用匿名内部类来模拟一些“真实”的闭包行为,但是它们并不能实现所有闭包的特性,因为 Java 本身不支持闭包。那又意味着什么呢?

列表 5 展示了一个闭包特殊之处的例子。它是使用 Groovy 来写的(Groovy 通过它的代码块机制来支持闭包)

```
defmakeCounter() {
defvery_local_variable = 0
return { return very_local_variable += 1 }
}
c1 = makeCounter()
c1()
c1()
c1()
c2 = makeCounter()
println "C1 = ${c1()}, C2 = ${c2()}"
// output: C1 = 4, C2 = 1
```

makeCounter ( )方法首先用一个合适的名字定义了一个本地变量,然后返回一个使用该变量的代码块。注意这里 makeCounter ( )方法返回的是一个代码块,而不是值。这个代码块除了增加本地变量的值并返回外什么都不做。我这里显式的调用 Return 是为了让代码看起来不那么神秘,其实在 Groovy 中 Return 并不是必须的。

为了运用 makeCounter ( )方法,我分配了一个代码块给 C1 变量,然后调用 3 次。这里我使用了 Groovy 的语法糖去执行一个代码块。下一步,我再次调用 makeCounter ( ),并分配一个实例给 C2。最后,我一起执行 C1 和 C2。注意每一个代码块都保存了对单独的局部变量的跟踪。这就是所谓的封闭上下文。尽管本地变量定义在方法内,但是代码块绑定了这个变量因为它引用了该变量,这就意为着只要代码块实例一直活着,代码块将一直跟踪该变量。

你所能看到最接近的 Java 版本就是列表 6 :

```
public class Counter {
private int varField;
public Counter(int var) {
varField = var;
    }
public static Counter makeCounter() {
return new Counter(0);
    }
public int execute() {
return ++varField;
    }
}
```

函数式思维 :思维的功能(二)VI

尽管 Counter 类里的几个变量是必要,但是你又很纠结于自己来管理状态。状态的初始(包括在多线程环境下,以一种忧心忡忡的心态来运行自己的代码),让语言或框架显式的为你管理状态。

我们最终将在下一个 Java 的发布里面(关于这个话题已经超出了这篇文章的讨论范围)获得闭包。它们出现在 Java 里有两个显式的好处 :

第一,它将极大的简化框架和库作者的能力需求。

第二,它将为所有语言(运行于 JVM 上的语言)的闭包提供一个底层的通用基础。

尽管很多的 JVM 语言都支持闭包,但是它们都实现了各自的版本,这也使得语言间传递闭包变的难以处理。

如果 Java 语言定义了一个单独的格式,那么所有其他语言都可以基于它来实现。

总结

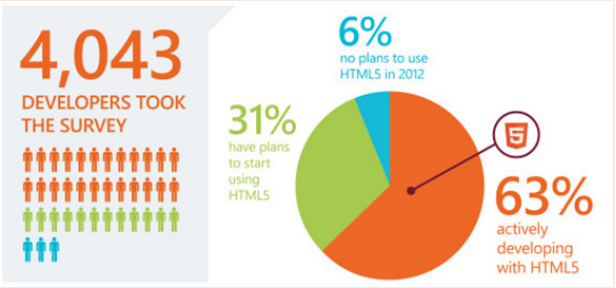
放弃对底层细节的控制将是软件开发领域的一个普遍趋势,我们应兴高采烈的放弃了这些职责(如垃圾收集,内存管理和硬件异构等)。

函数式编程是下一个抽象级别上的飞跃 :尽可能多的放弃那些平乏的细节如迭代遍历,并发执行和运行时状态控制。

但这并不意味着你无法控制代码,在必要的时候你还是可以拿回控制权的。

下一篇文章,我将继续探索 Java 里的函数式编程构造,并进一步介绍科里化和偏函数的应用。

专题:尘埃落定 W3C宣布HTML 5规范正式定稿



W3C:HTML5规范开发完成

HTML 5是一套开放的标记语言,主要由 W3C 和网络超文本应用技术小组(WHATWG)负责开发。目前该标准已包含超过 100 种不同规范,定义了下一代 Web 应用程序所需的技术。

王淮:HTML5的明天,局部有小雨

现在 HTML5 技术很热,那么作为开发者,你的产品是否可以赶个潮流,可以用 HTML5 来做呢?Facebook 前研发经理王淮(weibo)在博客……

HTML5欧亚冰火两重天

有评论说国人把HTML5捧的太高,HTML5的不成熟引发出各种问题,加上FACEBOOK的失败,似乎验证了HTML5注定没有好下场……

HTML5将会带来一场Web革命

一个不起眼的对网页的标记机制为何有这么大的影响? HTML5 引起的广泛关注是否仅仅是一时的科技狂热? 总之,为什么计算机专家需要关心这个呢?

专题 链接:
http://developer.51cto.com/art/201212/373250.htm